



Capstone Courseware, LLC

33 Boylston Street
Jamaica Plain, MA 02130

877-227-2477
capstonecourseware.com

Java Persistence Using Hibernate

Version 3.2.1

Instructor's Guide



Overview

Of all the persistence frameworks currently in the market, Hibernate has emerged as a clear favorite. This is partly due to the quality of the framework and partly due to the fact that the authors of the framework were highly influential in the design of the EJB3 and JPA specifications.

Despite the emphasis on server-side Java applications inherent in those specifications, this course treats Hibernate from a Java Standard Edition point of view. While discussions are included about running inside an application server with database pooling and container managed transactions, neither of those capabilities significantly affects the code. Only the configurations need to change and those in a relatively minor way.

Hibernate, as an open source project, is actively growing and changing. Its merger with the EJB and JPA specifications may slow that process a bit, so we may have entered a period of relative stability for the API. The latest released version of the framework, 3.2, is compatible with those specifications and is therefore the version used in this course. Interestingly enough, however, the changes in the API since version 3.0 have been minimal, so users familiar with 3.0 or 3.1 will likely feel right at home.

The materials assume that we are using the Java 5 standard edition (also known as version 1.5, depending on which web page at Sun you consult) for the client code. This was done largely for convenience and in anticipation of future developments. The fact remains that the dominant application server in the current marketplace (IBM's WebSphere Application Server) does not support Java 5. Since the only aspect of Hibernate that requires Java 5 is the annotations capability, this is not discussed in this course. It is expected that a future module will address this issue as well as JPA integration.





Timeline

The following breakdowns are approximate, and every class will vary.

Day 1

| | |
|---------|-----------|
| 2 hours | Chapter 1 |
| 1 hour | Chapter 2 |
| 3 hours | Chapter 3 |

Day 2

| | |
|----------|-----------|
| 4 hours | Chapter 4 |
| 1½ hours | Chapter 5 |
| 1 hour | Chapter 6 |

Since Hibernate is all about object/relational mapping, the material needs to address both object-oriented design using Java and database schema design issues. Most students will be comfortable with one or the other but not both, so both are reviewed as necessary. Some of the books in the literature start with Java classes and create schemas to match, but in our experience that the reverse – working from an existing schema – is much more common. As a result, in both the major case studies addressed in these materials a database schema is assumed and Hibernate models are created to model it.

The first schema is an almost trivial many-to-many mapping of Users to Roles, such as might occur in a simple security system. The Tomcat application server uses a system like this. The development of the mapping documents for this schema is covered in Chapters 1 through 3.

The other major schema modeled in these materials is called Earthlings and involves four tables representing employees, departments, jobs, and locations. Modeling this schema occupies the rest of chapter 3 and all of Chapter 4, which allows us to examine different types of association and inheritance relationships in Hibernate.

Chapters 5 and 6 cover the two major querying APIs, Criteria and HQL. It is likely that students will be very interested in both, which is fortunate since they are both powerful and quite straightforward. Hopefully, by structuring the materials so that we end on those two topics, the students will feel the course ends on a high note.





Tools Deployed with the Lab Software

This course's software requires separate setups of a Java SDK, the Crimson text editor, and/or Eclipse 3.x; otherwise it is self-contained, as the lab installer sets up not only the lab software but necessary tools: Ant, Hibernate, and MySQL.

The Hibernate deployment in `c:\Capstone\Tools\Hibernate3.2` is a subset of the full 3.2 release. That distribution is quite large, some 24meg when ZIPPed, and we're trying to balance the convenience of having everything pre-deployed in the classroom with the need for reasonably small downloads. So for Hibernate, we've taken the original image and cut out several large pieces: the source code and test suites, the optional JARs, and all but a small slice of the API documentation. This will probably go completely unnoticed by students, but if they use the API docs they may well find that there are blank patches on the fringes of the docs they want to read. If a student is curious and wants the full Hibernate API docs, or perhaps the source code, just direct him or her to the public download of the full distribution, as documented in the coursebook.

MySQL is a similar story: what we deploy in `c:\Capstone\Tools\MySQL5.0` is not the full distribution, but a tiny kernel that's enough for our purposes: basically the `mysqld` server, the `mysqladmin` utility, the `mysql` terminal, and supporting libraries and administrative databases.

(Of course, if students are interested in using a different RDBMS this will come up fairly frequently, this should be easy to do – as easy as JDBC makes it, meaning change the DB URL, driver class, credentials, and any other vendor-specific properties. These are all captured in a Hibernate configuration file, which means they'll need to be updated in each step of each case study or each demo or lab directory. It's not trivial but it's fairly painless.)



Ant Build Process

Though most students will be happy to let the **ant** command take care of things, some students (and most instructors) will want to understand the inner workings here a little better. Each project in the course has its own **build.xml** and **build.properties** files; these rely on the master environment variable **CC_MODULE** to import targets and properties in a central directory, **%CC_MODULE%\Ant** which is typically **c:\Capstone\Hibernate\Ant**. Information here defines a routine for building a Hibernate application.

For the most part, the Ant build just compiles Java classes and makes sure that Hibernate and the JDBC driver are in the class path. The provided **run.bat** script simplifies the launching of a target application class; it just wraps a call to **ant** with the target **test** and application class and command-line arguments defined as properties.

The one less obvious trick involves the configuration and mapping files. These need to be colocated with the class files, and so we need to copy them from the **src** tree to the **build** tree. That's simple enough, but to make sure that any changes to these XML files are immediately reflected in the next test, this file copy is repeated just prior to running any application: the **test** target depends on **copy-config** and then **run**.





Eclipse Overlays

Capstone Courseware provides an optional package of workspace and project files for Eclipse 3.2 for this course. (See the course Setup Guide for download URLs.) Instructors, use this package on your own initiative and at your own risk. You should have experience yourself with Eclipse before using the overlay package in the classroom. The workspace and projects have been tested lightly with the course but are not part of the standard product.

That said, this overlay should save a good deal of work for those who wish to use Eclipse instead of the text editor and command-line tools that are standard for the course. See the documentation inside the workspace itself for general notes on usage: when you first open the workspace you'll see a **ReadMe.txt** that directs you to deeper, HTML-based documentation. Be prepared to walk students through the first few exercises in Eclipse; the notes in this file are for experienced Eclipse users, and will not be clear to many students on their own.

See also the file **Hibernate Module Notes.html** (available in the IDE as well) for specific matters related to using Eclipse WTP with this course.

Again, Capstone Courseware can only offer complete technical support on the standard course, and while we hope this overlay is convenient, it is not as thoroughly tested as the core lab image at this time. If a given exercise is giving trouble, please be certain to build and run it from the command line, using the SDK tools as prescribed in the student guide, before contacting Capstone.





Teaching Notes

Chapter 1

This chapter introduces the concept of object/relational mapping in general and the Hibernate framework in particular. Rather than go into depth here, however, a single very simple class is mapped as a kind of “Hello, World!” example. This will hopefully allow the students to get up and running quickly, as well as demonstrating all the different moving parts in a Hibernate application.

Chapter 2

Here we cover background material related to Hibernate configuration. Hibernate can be configured using an XML file, a properties file, and programmatically. All three mechanisms are discussed in this chapter. Even though the XML configuration is the most common, the properties file is of particular interest because a highly commented version of this file is available in the Hibernate distribution. This provides a convenient basis for a discussion of different Hibernate capabilities.

In many open source projects, the distribution is as much or more helpful than the available documentation. Hibernate's distribution comes with sample configuration files, a sample application, a sample logging file, a PDF reference document, the JavaDoc API, a well-written tutorial, and the complete source code. These are invaluable resources for any developer planning to use Hibernate, so we spend a fair amount of time discussing them here.

Chapter 3

In this chapter, the primary Hibernate classes and interfaces are discussed, as well as common idioms for their usage. The emphasis is not on the mapping documents, but rather on using Hibernate in client-side code effectively. This includes discussions of what is meant by a Hibernate session and how it relates to transactions.

The chapter also includes a discussion of the Data Access Object design pattern, which is used throughout the rest of the course.

The discussion of object states (persistent, detached, and transient) is fundamental to understanding the framework and should be emphasized.





Chapter 4

This chapter returns to the subject of mapping relational tables into classes. After reviewing object-oriented design and contrasting it with relational modeling, the chapter discusses how to model various types of association, as well as inheritance. This chapter also goes through the detailed structure of the mapping documents.

Chapter 5

This chapter discusses the Criteria Query API, which provides an object-oriented way of building queries. This capability is both powerful and easy for an object-oriented developer to understand.

Chapter 6

HQL, the Hibernate Query Language, is examined in this chapter. HQL is an object-oriented SQL variant, so it is important to emphasize that even though the queries look like SQL, the names of classes appear in the queries, not table names. Students normally find the absence of a select clause surprising, which is an excellent way of emphasizing that we're retrieving objects, not just fields.

Many IDE's (MyEclipse and JBoss IDE to name two) include the ability to type in HQL queries and see their results right away. In our case we can't assume that capability, so the HQLQueries class shows how to make small methods to execute test queries.





Revision History

Version 3.2.1 fixes a handful of typos and omissions in the book and a small bug in one of the labs. Also the Eclipse overlay is updated to a new structure that greatly simplifies testing the exercises from within the IDE.

Version 3.2 is the initial release of the course.





Troubleshooting

If you run into any trouble with code exercises, the first and best thing to do is to double-check that the classroom machines have been set up precisely according to the course setup guide. Especially, the wrong version of a tool can cause significant problems; don't wander off-book in this way unless absolutely sure you can support the software that you prefer and that we haven't tested. Check environment variable settings carefully, too; these are the cause of a great many classroom glitches.

Below are some specific pitfalls that have come up in previous offerings of the course:

- Since Java code is easier to refactor than XML mapping documents, be sure to check that the XML mapping files are found by the framework. The first place to check is always the mapping elements in the **hibernate.cfg.xml** file. Make sure they point to the proper locations.
- Hibernate comes with a very large number of JAR files, so classpath issues can arise if you're not careful. Particularly annoying is that the **hibernate3.jar** file is not located in the lib directory with the rest of them, so watch for that.
- It's a simple thing, but the DTD for the Hibernate configuration file is different from the DTD for the mapping files. Be sure each file is using the proper DOCTYPE declaration.
- When unexpected behavior arises, it's always good to turn on SQL logging. The distribution comes with a sample **log4j.properties** file that has commented-out versions of many convenient settings. Also, enabling the **show_sql** and optionally the **format_sql** property will help students with SQL experience considerably.





Errata

At this time there are no errata for the course.





Feedback

We very much appreciate whatever feedback we can get on our courseware – especially from the instructor's perspective. Naturally, the more specific, the better, and we strongly encourage you to make notes on issues you may encounter in the classroom, whether they're typos, missing files, or suggestions for clearer language to explain a concept. We can't guarantee that we'll act on every suggestion, but we're aggressive about stamping out problems and try to be highly responsive. Hopefully this means that when you give us good feedback, you get a better course the next time you need to teach it.

Please direct all courseware feedback to

Will Provost
Capstone Courseware
<mailto:provost@capcourse.com>
877-227-2477

For anyone who's interested, we have a very informal defect-tracking system, based in Excel spreadsheets with columns to capture defect location, nature, status, and author feedback. Ultimately, feedback goes into these sheets, so if you want a template, we'll be happy to provide one, to facilitate the reporting process.

